平面几何

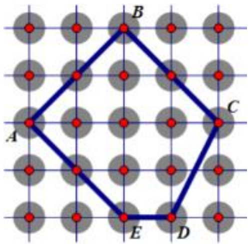
2019年5月30日　　20:46

一. 向量

1. 例题：格点pick岛的岛内格点数n（CF101873G）

    i. 引：pick定理：S=顶点在格点上的任意多边形面积，b=边上格点数，n=边内格点数，则2S=2n-b-2，即n=(2S-b)/2+1

    ii. pick定理的主观理解：以格点为正中心，画出一个“棋盘格”，想象每个红点向外散发出一个黑色菱形，则多边形内每有一个格点，就代表有一个菱形被整个包含；多边形边上的格点代表的菱形只有一半被多边形包含；顶点代表的菱形只有<一半被包含，不过利用多边形内角和=角数*180-360，可知顶点代表的菱形的面积和恰好是一整个格点，所以要-1

    iii. 

    iv. 2S可以利用向量点积求，从多边形内或多边形边上任一点出发，顺时针或逆时针遍历每对相邻顶点，求以两向量为邻边的平行四边形的和即可（题目保证以顺时针给出各顶点）

    v. b可以遍历每条边，累积边在xy轴上的投影的gcd

```
struct Point{
    int x, y;
    Point(int x=0, int y=0): x(x),y(y){    }
    Point operator-(Point r){
        return Point(x-r.x, y-r.y);
    }
};
ll cross(Point l, Point r){        //向量叉乘数量积
    return ll(l.x)*r.y - ll(l.y)*r.x;
}
ll area(Point a, Point b, Point c){      //三角形abc的面积的两倍（可能为负）
    return cross(b-a, c-a);
}

Point p[MN];
inline int cal_b(Point l, Point r){      //求线段lr上的格点数
    return __gcd(abs(l.x-r.x), abs(l.y-r.y));
}
```

    vi.

```
for(int i=0; i<n; ++i)
    cin>>p[i].x>>p[i].y;
ll s=0;
for(int i=1; i<n; ++i)      //以p[0]为中心点分割多边形
    s+= area(p[0], p[i-1], p[i]);
s=abs(s);     //注意s可能是负的
ll b=cal_b(p[0], p[n-1]);
for(int i=1; i<n; ++i)
    b+=cal_b(p[i], p[i-1]);
```

```
    cout<< (s-b) /2 +1;
```

2. 例题：顺时针给出n个点，求m个点是否都在其围成的多边形内（CF166B）
    i. 因为保证是顺时针给出的，所以可以二分顺时针旋转过程中第几个包含0号点三角形包含样例点，再判断一下该点是不是恰在三角形中即可
    ii. 具体的判断，可以通过向量叉积正负号，详见程序

```
int x[MN],y[MN];    //题目保证外多边形点按顺时针给出
inline ll xcross(int a,int b,int c){   //ab叉乘ac
    return ll(x[b]-x[a])*(y[c]-y[a])-ll(x[c]-x[a])*(y[b]-y[a]);
}
inline bool collinear(int a,int b,int c){    //abc是否共线
    return xcross(a,b,c)==0;
}
inline bool clockwise(int a,int b,int c){    //abc是顺时针
    return xcross(a,b,c)<0;
}
inline bool counterclockwise(int a,int b,int c){    //abc是逆时针
    return xcross(a,b,c)>0;
}
int n,m;
bool chk(){  //检查新输入的x[n],y[n]是否出现在了前n个点里
    if(collinear(n,0,1)||collinear(n,0,n-1)) return 0;
    int l=0, r=n-1, mid, a;   //a存取最优mid
    while(l<r){  //0为源点，1开始顺时针转，转超过n之前最后一个mid
        mid=l+r>>1;  //设0点在左下角，顺时针说明mid在直线0n右
        if(clockwise(n,mid,0)) r=mid;
        else a=mid, l=mid+1;
    }      //现在直线0a和直线0{a+1}之间应有n点
    if(!clockwise(n,a,a+1)) return 0;      //n在边上或图形外
    return 1;
}
```

3. 例题：关于判断点在多边形内（CF101623G）
    i. 题意：一个顶点在x轴的外正k边形减去内正边形恰能包住所有给定点，中心在坐标原点，求这种正k边框（3~8）的面积比值
    ii. 面积大小与在x轴的点的x坐标的平方有关
    iii. 二分该点x坐标，用向量叉乘正负来判断是否在多边形内

```
const db eps = 1e-7;
const db pi = acos(-1);
int x[MN], y[MN];  //空开下标0
db px[10],py[10];  //多边形顶点
struct vec{  //二维向量
    db x,y,x0,y0;//向量，起点
    vec(db x=0,db y=0,db x0=0,db y0=0):x(x),y(y),y0(y0),x0(x0){}
}v[10];
db cross(vec a,vec b){    //向量叉积的数量值
    return a.x*b.y - a.y*b.x;
}

int n;
inline int chk(int k,db r){      //边数k，半径r，返回能包住几个点
    int cnt=0;   //能包住的点
```

```
    for(int i=1; i<=k; ++i)   //逆时针
        px[i]=r*cos((i-1)*2*pi/k),
        py[i]=r*sin((i-1)*2*pi/k);
    for(int i=1; i<k; ++i)
        v[i]=vec(px[i+1]-px[i], py[i+1]-py[i], px[i], py[i]);
    v[k]=vec(px[1]-px[k], py[1]-py[k], px[k], py[k]);
    for(int i=1; i<=n; ++i){  //遍历各给定点
        int j=1;      //遍历正在chk的边
        for(; j<=k; ++j)    //定点指向边的起点 叉乘 逆时针的边
            if(cross(vec(x[i]-v[j].x0, y[i]-v[j].y0), v[j]) > eps)
            break;
        if(j>k)
            ++cnt;
    }
    return cnt;
}

inline void solve(){
    cin>>n;
    for(int i=1; i<=n; ++i)
        cin>>x[i]>>y[i];
    int ans=0; db ans2=0;
    for(int k=3; k<=8; ++k){
        db l=eps, r=1e9;
        while(l+eps<r){
            db mid=(r+l)/2;
            if(chk(k,mid)==n) r=mid;
            else l=mid;
        }
        db r1=l;
        l=eps, r=1e9;
        while(l+eps<r){
            db mid=(l+r)/2;
            if(chk(k,mid)) r=mid;
            else l=mid;
        }
        db r2=l;
        if(r2/r1>ans2) ans=k, ans2=r2/r1;
    }
    cout<<ans<<' '<<fixed<<setprecision(10)<<ans2*ans2;
}
```

二. 圆

1. 例：与x轴相切且包含所有给定点的最小圆半径（CF1059D）
    i. 首先给定点不同号肯定不行（不与x轴相切）然后为了方便把负的取正
    ii. 之后二分圆的半径长度rd，依次check能包含各点的x的范围（解方程得rd方-(rd-y)方，x±其平方根即为可行圆心的横坐标范围）有交集即可行
    iii. 上式容易忽略小数的精度，可以用平方差公式缩小指数差距来减少精度问题

```
bool chk(ld rd){
    ld l=-1e17, r=1e17;
    for_(i,0,n){
        if(y[i]>rd*2) return 0;
        ld t= sqrt(y[i])* sqrt(rd+rd-y[i]);
        l=max(l, x[i]-t);
        r=min(r, x[i]+t);
    }
    return l<r;  //圆心横坐标交集非空
}
```

2. 例：大圆内切的不相交小圆数

i. 极限情况是所有小圆都相切，此时所有小圆都相切，且大圆心到切点的垂线段恰为中垂线，此时一个小圆占了两倍asin(double(r)/(R-r))大圆心角的区域，计算360°有几个这样的圆心角即可

ii. 麻烦的是精度问题和arcsin（）值域不能>1的特判，以及，例如6 9 3，可能会卡出5.99999这样

```
int n,R,r;
cin>>n>>R>>r;

if(r>R){
	cout<<"No";
	return 0;
}
if(n==1){		// && r<=R
	cout<<"Yes";
	return 0;
}
if(r>R/2){		// && n>1
	cout<<"No";
	return 0;
}

//	cout<<asin(double(r)/(R-r))<<" ";
	cout<<int(3.1416/asin(double(r)/(R-r)))<<" ";
	int t = int(/*0.000001 + */3.1416/asin(double(r)/(R-r)));
	if(n<=t)
		cout<<"Yes";
	else
		cout<<"No";
```

三. 向量和面积模板

1. 结构体模板from刘汝佳，不过被我魔改成了面向对象模式......

i.
```
int dcmp(double x){	//三态函数返回符号
	if(fabs(x) < 1e-6)
		return 0;
	else
		return x<0 ? -1 : 1 ;
}
```

ii.
```
struct point{
	double x, y;
	point(double x, double y): x(x), y(y) {}	//构造
	double dot(point r){	return x*r.x + y*r.y;	}		//点乘
	double length(){		return sqrt(dot(*this));	}
	double angle(point r){
		return acos( dot(r) / length() / r.length());		}
	double cross(point r){	return x*r.y - y*r.x;	}		//叉乘
	double area(point m ,point r){
		return point(m.x-x, m.y-y).cross(point(r.x-x, r.y-y));
	}		//平行四边形面积
	point rotate(double rad){
		return point(x*cos(rad) - y*sin(rad), x*sin(rad) + y*cos(rad));
	}		//逆时针旋转90°
```

```
};
```

2. #include<complex>
   i. `typedef complex<double> point;`
   ii.
   ```
   double dot(point l, point r){
       return real(conj(l)*r);
   }      //共轭l * r 的实部相当于lr点积
   ```
   iii.
   ```
   double cross(point l, point r){
       return imag(conj(l)*r);
   }      //共轭l * r 的虚部相当于lr叉积
   ```
   iv.
   ```
   point rotate(point l, double r){
       return l* exp(point(0, r));
   }
   ```

四. 公式

1. 海伦公式：S=√p(p-a)(p-b)(p-c)其中p为半周长
2. 积化和差

   $\sin\alpha \cdot \cos\beta = (1/2)[\sin(\alpha+\beta)+\sin(\alpha-\beta)]$

   $\cos\alpha \cdot \sin\beta = (1/2)[\sin(\alpha+\beta)-\sin(\alpha-\beta)]$

   $\cos\alpha \cdot \cos\beta = (1/2)[\cos(\alpha+\beta)+\cos(\alpha-\beta)]$

   $\sin\alpha \cdot \sin\beta = -(1/2)[\cos(\alpha+\beta)-\cos(\alpha-\beta)]$
3. 和差化积
   i.
   $$\sin\alpha \pm \sin\beta = 2\sin\frac{(\alpha\pm\beta)}{2}\cos\frac{(\alpha\mp\beta)}{2}$$
   $$\cos\alpha + \cos\beta = 2\cos\frac{\alpha+\beta}{2}\cos\frac{\alpha-\beta}{2}$$
   $$\cos\alpha - \cos\beta = -2\sin\frac{\alpha+\beta}{2}\sin\frac{\alpha-\beta}{2}$$